Free Questions for CKS by certsdeals

Shared by Huff on 05-09-2022

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

SIMULATION

A container image scanner is set up on the cluster.

Given an incomplete configuration in the directory

/etc/Kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://acme.local.8081/image_policy

1. Enable the admission plugin.

2. Validate the control configuration and change it to implicit deny.

Finally, test the configuration by deploying the pod having the image tag as the latest.

Options:

A) Send us the Feedback on it.

Answer:

A

Question 2

Question Type: MultipleChoice

SIMULATION

Create a network policy named allow-np, that allows pod in the namespace staging to connect to port 80 of other pods in the same namespace.

Ensure that Network Policy:-

1. Does not allow access to pod not listening on port 80.

2. Does not allow access from Pods, not in namespace staging.

Options:

A) Explanation:
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: network-policy
spec:
podSelector: {} #selects all the pods in the namespace deployed
policyTypes:

- Ingress
ingress:
- ports: #in input traffic allowed only through 80 port only
- protocol: TCP
port: 80

Answer:

A

Question 3

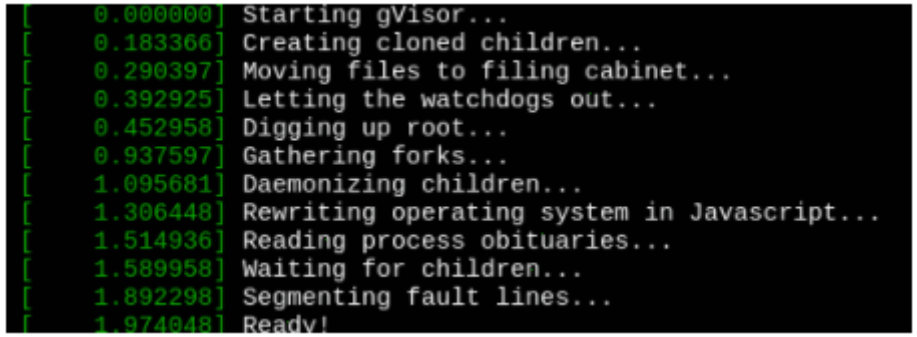
Question Type: MultipleChoice

SIMULATION

Create a RuntimeClass named untrusted using the prepared runtime handler named runsc.

Create a Pods of image alpine:3.13.2 in the Namespace default to run on the gVisor runtime class.

Verify: Exec the pods and run the dmesg, you will see output like this:-



Options:

A) Send us your feedback on it.

Answer:

A

Question 4

Question Type: **MultipleChoice**

SIMULATION

Before Making any changes build the Dockerfile with tag base:v1

Now Analyze and edit the given Dockerfile(based on ubuntu 16:04)

Fixing two instructions present in the file, Check from Security Aspect and Reduce Size point of view.

Dockerfile:

```
FROM ubuntu:latest

RUN apt-get update -y

RUN apt install nginx -y

COPY entrypoint.sh /

RUN useradd ubuntu

ENTRYPOINT ["/entrypoint.sh"]

USER ubuntu
```

entrypoint.sh

```
#!/bin/bash

echo "Hello from CKS"
```

After fixing the Dockerfile, build the docker-image with the tag base:v2

To Verify:Check the size of the image before and after the build.

Options:

A) Send us the Feedback on it.

Answer:

A

Question 5

Question Type: **MultipleChoice**

SIMULATION

Enable audit logs in the cluster, To Do so, enable the log backend, and ensure that

1. logs are stored at /var/log/kubernetes/kubernetes-logs.txt.

2. Log files are retained for 5 days.

3. at maximum, a number of 10 old audit logs files are retained.

Edit and extend the basic policy to log:

1. Cronjobs changes at RequestResponse

2. Log the request body of deployments changes in the namespace kube-system.

3. Log all other resources in core and extensions at the Request level.

4. Don't log watch requests by the "system:kube-proxy" on endpoints or

Options:

A) Send us the Feedback on it.

Answer:

A

Question 6

Question Type: **MultipleChoice**

SIMULATION

Given an existing Pod named nginx-pod running in the namespace test-system, fetch the service-account-name used and put the content in /candidate/KSC00124.txt

Create a new Role named dev-test-role in the namespace test-system, which can perform update operations, on resources of type namespaces.

Create a new RoleBinding named dev-test-role-binding, which binds the newly created Role to the Pod's ServiceAccount (found in the Nginx pod running in namespace test-system).

Options:

A) Sendusyourfeedbackonit

Answer:

A

Question 7

Question Type: **MultipleChoice**

SIMULATION

Create a PSP that will only allow the persistentvolumeclaim as the volume type in the namespace restricted.

Create a new PodSecurityPolicy named prevent-volume-policy which prevents the pods which is having different volumes mount apart from persistentvolumeclaim.

Create a new ServiceAccount named psp-sa in the namespace restricted.

Create a new ClusterRole named psp-role, which uses the newly created Pod Security Policy prevent-volume-policy

Create a new ClusterRoleBinding named psp-role-binding, which binds the created ClusterRole psp-role to the created SA psp-sa.

Hint:

Also, Check the Configuration is working or not by trying to Mount a Secret in the pod maifest, it should get failed.

POD Manifest:

apiVersion: v1

kind: Pod

metadata:

name:

spec:

containers:

- name:

image:

volumeMounts:

- name:

mountPath:

volumes:

- name:

secret:

secretName:

Options:

A) Explanation:
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
name: restricted
annotations:
seccomp.security.alpha.kubernetes.io/allowedProfileNames: 'docker/default,runtime/default'
apparmor.security.beta.kubernetes.io/allowedProfileNames: 'runtime/default'
seccomp.security.alpha.kubernetes.io/defaultProfileName: 'runtime/default'

```
apparmor.security.beta.kubernetes.io/defaultProfileName: 'runtime/default'
spec:
privileged: false
# Required to prevent escalations to root.
allowPrivilegeEscalation: false
# This is redundant with non-root + disallow privilege escalation,
# but we can provide it for defense in depth.
requiredDropCapabilities:
- ALL
# Allow core volume types.
volumes:
- 'configMap'
- 'emptyDir'
- 'projected'
- 'secret'
- 'downwardAPI'
# Assume that persistentVolumes set up by the cluster admin are safe to use.
- 'persistentVolumeClaim'
hostNetwork: false
hostIPC: false
hostPID: false
runAsUser:
# Require the container to run without root privileges.
rule: 'MustRunAsNonRoot'
seLinux:
# This policy assumes the nodes are using AppArmor rather than SELinux.
```

rule: 'RunAsAny'
supplementalGroups:
rule: 'MustRunAs'
ranges:
Forbid adding the root group.
- min: 1
max: 65535
fsGroup:
rule: 'MustRunAs'
ranges:
Forbid adding the root group.
- min: 1
max: 65535
readOnlyRootFilesystem: false

Answer:

A